Computing a Network of ASRs using a Mobile Robot Equipped with Sonar Sensors

Chee K. Wong, Wai K. Yeap, and Jochen Schmidt
Institute for Information Technology Research
Auckland University of Technology
Auckland, New Zealand
{chee.wong, wai.yeap, jochen.schmidt}@aut.ac.nz

***Abstract*— This paper presents a novel algorithm for computing absolute space representations (ASRs) [1]-[2] for mobile robots equipped with sonar sensors and an odometer. The robot is allowed to wander freely (i.e. without following any fixed path) along the corridors in an office environment from a given start point to an end point. It then wanders from the end point back to the start point. The resulting ASRs computed in both directions are shown.**

***Keywords*—robotics, cognitive mapping**

I. INTRODUCTION

In recent years the idea of computing a network of local (metrical) spaces as a representation of the environment that an autonomous agent has experienced is gaining momentum. This idea, commonly known as computing a hybrid map, is favored by both the pragmatists (i.e. those researchers interested in building mobile robots) and the theorists (i.e. those researchers interested in developing computational theories of cognitive maps).

Thrun [3] implemented a mobile robot that computes grid-based maps of its environment using artificial neural networks and naïve Bayesian integration. From the grid-maps, it generates networks of empty spaces. The network representation is used for fast planning and problem solving. Tomatis, Nourbakhsh and Siegwart [4] offered a different solution – they compute a topological network of open spaces found in corridors and metric maps for rooms. The latter are then attached to the appropriate nodes in the network. Open spaces in corridors are distinguished by the presence of physical corners separating them.

Kuipers and his team [5]-[6] recently investigated the use of hybrid maps within the Spatial Semantic Hierarchy framework [7] for cognitive mapping. They showed how a metric map constructed for each local environment (which they called local perceptual maps) is analyzed to generate a local topology description which is then used to create a network of "places".

In this paper, we present a different algorithm for computing a hybrid map. The algorithm is specifically designed for use on a mobile robot equipped with sonar sensors and an odometer. Our approach and motivation in constructing a hybrid map is different from those mentioned earlier. For example, in Thrun's approach, the network is computed after having computed a detailed metric map of the environment. The network is computed because the metric map is found to be inefficient for planning purposes. The network is thus an abstraction of the information made explicit in the metric maps; it is not computed directly from the environment. In contrast, Tomatis et al. computed the network representation first and the representation played an important role in learning the environment (especially in solving the localization problem). However, their implementation is restricted to an environment consisting of offices and corridors and the environment must be learned by exploring corridors first and then offices.

Kuipers and his team introduced the "perceptual maps" to ground the place representations described in his Spatial Semantic Hierarchy. Their work is most similar to ours in that we both are attempting to ground a theory of cognitive mapping using a mobile robot. However, instead of generating a local topology for each metric map, our metric map is constructed only to provide a rough description of the shape of the local environment. It has been argued that such a representation forms the basis for subsequent learning of a much richer description of each local environment [1]-[2]. Following [1]-[2], we continue to refer to such a representation as an absolute space representation (ASR). Our hybrid map is thus a network of ASRs; a link between two ASRs in the network indicates that the individual has experienced moving between the two local environments.

It is important to realize that computing a hybrid map consisting of a network of ASRs does not enable one, in general, to then use the map to move about successfully in the environment, at least not initially. To do so, it requires the individual to have successfully localized itself while computing the map. For humans/animal cognitive mapping, this localization problem is solved via repeated visits to a place and via the enriching of each local environment with more information other than improving the precision of the metric map. Our primary goal is to develop a robotic platform for testing theories of cognitive mapping. We are not just interested in designing a robot that can compute a map of its environment successfully for navigation purposes.

In this paper, we show how a robot equipped with sonar sensors and an odometer can compute a network of ASRs. Yeap and Jefferies [2] showed the importance of identifying exits when computing ASRs. Exits are important because through them, one explores the world. However, with the use of sonar sensors, it is difficult for our robot to locate exists

1-4244-0025-2/06/$20.00 ©2006 IEEE

reliably. Furthermore, given the poor readings, it is also better to compute an ASR after moving through it rather than computing one at the entrance of a new ASR. Our robot is thus more like a blind person than a sighted person.

A new algorithm is proposed. The key ideas underlying our new algorithm are as follows:

1. ASRs are computed for each path traversed – a path is a single continuous movement of the robot through the environment (i.e. without any stopping or turning);
2. The important exits found in a path are the exits at both ends of it (i.e. given the poor sensing, one cannot trust the side exits detected). This means that the required ASR for path is the bounded region for the path;
3. To compute the bounded region, preference is given to using the large surfaces as opposed to the smaller ones;
4. A split and merge algorithm [8] is then used to split or combine ASRs obtained from single paths to produce the final ASRs for the environment.

Section II describes the experiments we conducted to test our new algorithm. Section III presents the news algorithms in detail. Section IV presents the results obtained and section V concludes the paper with a discussion of future work.

II. THE EXPERIMENT

The robot we use is a Pioneer 2 robot from ActivMedia and it came with a ring of 8 sonar sensors. The robot is positioned somewhere in the corridor in an office environment and is allowed to explore the environment until it is told to stop. No modification of the environment is done. That is, things that already existed in the environment (such as rubbish bins, flower pots, cabinets, etc.) remain there and doors leading into offices are close or open depending on the time of the experiment.

The environment used and one of the paths the robot took is as shown in Fig. 1. It does not use a wall-following procedure to navigate. It simply moves forward until it could not and then it "looks" for an empty space to move forward again. "Looking" is done using all the eight sensors but information about the environment is sensed via the two side sensors. The exploration algorithm can now be described as follows:

1. Move in a "straight" line and collect sonar data from the sides;
2. Stop when an obstacle is encountered; and
3. Turn away from the obstacle but maintain a forward-going direction

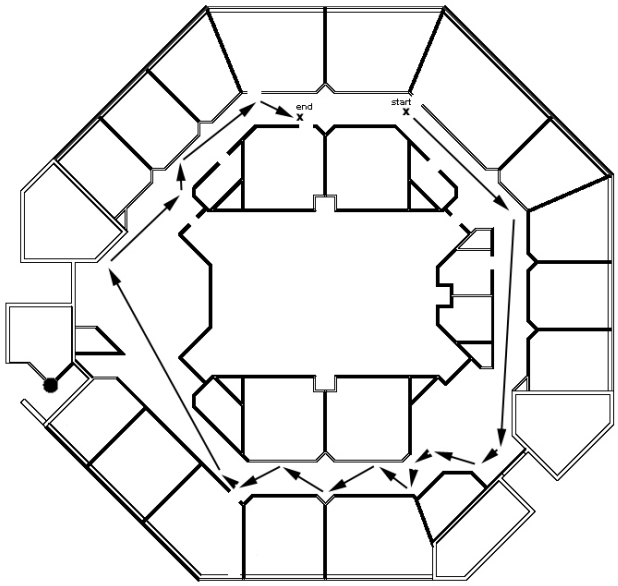

Figure 1. The environment used for the experiment and the path shown is one of the paths taken by the robot. The total distance traveled is about 70m.

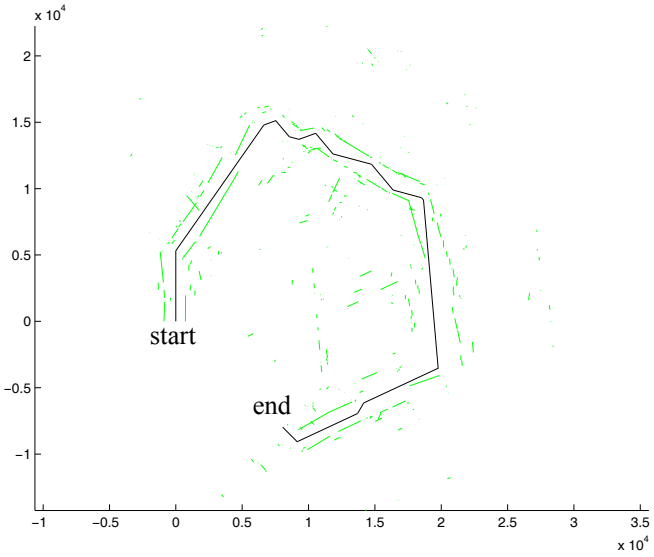

Figure 2. Initial surfaces perceived

III. ALGORITHMS FOR COMPUTING ASRS

Fig. 2 shows the initial surfaces computed from the sonar readings obtained by the robot traversing the environment along the path as shown in Fig. 1. As can be seen, there are too many spurious surfaces. We pre-process the input by removing outliers and insignificant surfaces. Fig. 3 shows the actual surfaces used as input for computing ASRs.

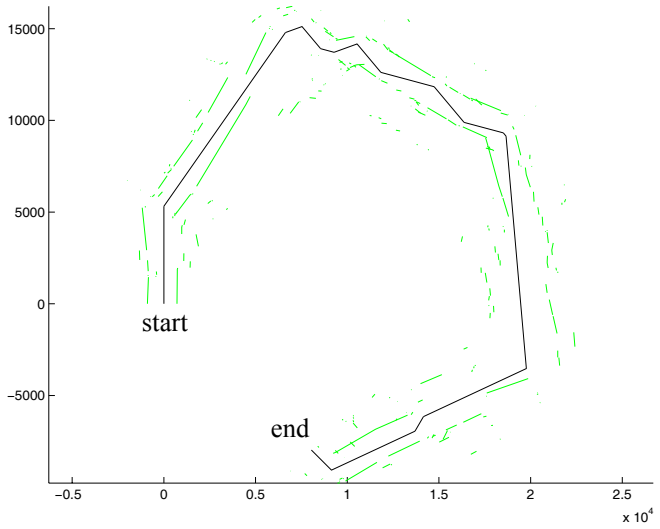

Figure 3. Sonar readings used as input for computing ASRs

The algorithm for computing ASRs consists of two steps. First it needs to compute an ASR after every single path traversed and then split or combine these ASRs after completing the journey to generate the final ASRs for the whole environment traversed.

Fig. 4 shows four paths the robot took to travel down a corridor. In the final analysis, the four ASRs produced will be merged into a single ASR. This single ASR represents the ASR computed for the whole corridor. ASRs computed for a single path needs to be merged or split later because the environment is sensed in a piecewise manner. Note that even for a robot with vision or with more powerful sensing, these two steps are also essential: the first step would compute an initial ASR from the first view after entering a new local environment and the second step would require updating the initial ASR with information obtained later from subsequent views [2]. Fig. 5 shows a situation where a split is required.

To compute an ASR for a path, we first identify those surfaces that could be part of the boundary for that path. Then we compute the boundary itself. Since with sonar, one has more confidence sensing the presence of a large surface than that of a small surface, it is thus best to consider the larger surfaces first. Furthermore, the large surfaces give one a strong sense of where the boundary lies.

To select the surfaces for boundary consideration, we begin with the surfaces that are considered as "large". If there are a sufficient number of them, we will use them to compute the boundary. If not, other smaller surfaces are also used. The algorithm that we have implemented can now be described as follows (the threshold values used are intuitively chosen):

Algorithm #1: Computing an ASR for a path

Step 1: selecting surfaces – four possible iterations:

1. Select all perceived surfaces that are greater than 700mm in length. If the sum of the surfaces selected is greater than 70% of the distance traveled, then go to step 2.

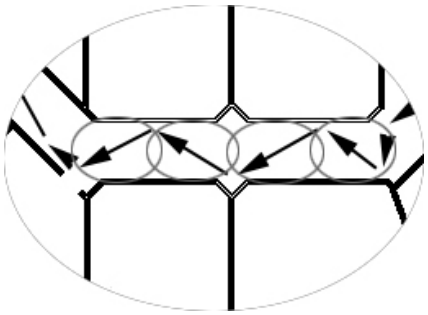

Figure 4. The four inner circles mark the four paths the robot took to travel down the corridor.

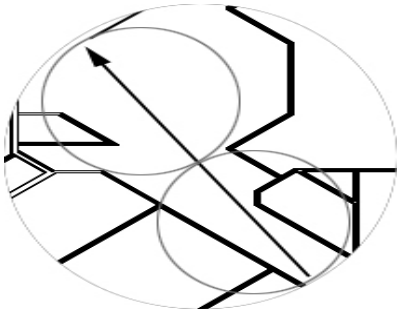

Figure 5. In this path, the robot generates a single ASR but in the final analysis, this ASR will be split into two ASRs. The two circles indicate where the split could happen.

2. Select all perceived surfaces that are greater than 500mm in length. If the sum of the surfaces selected is greater than 70% of the distance traveled, then go to step 2.
3. Select all perceived surfaces that are greater than 300mm in length. If the sum of the surfaces selected is greater than 70% of the distance traveled, then go to step 2.
4. Select all perceived surfaces that are greater than 200mm in length (those less than 200mm are considered as too small for consideration). Go to step 2.

Step 2: Given the surfaces from (1), compute the boundary for the ASR. The idea here is to "smooth" those surfaces that are close together and leave those that are separated by a "significant" gap as they are.

The next step is to split and merge the ASRs computed for the whole journey. This algorithm has been described in detail in [8]. Here we present a formal description of the algorithm:

Algorithm #2: Computing final ASRs after completing a journey – a split and merge algorithm.

1. Start with an initial set of points P^0, which consists of n_0 parts, $P^0 = \{P^0_0, \ldots, P^0_{n0-1}\}$. Each part P^0_i is

approximated by a function from F. Compute the initial residual error ε^0_i for each part of P^0.

2. **Split** each part P^k_i where $\varepsilon^k_i > \varepsilon_1$ into two parts P^{k+1}_j and P^{k+1}_{j+1}, compute the approximation and residuals ε^{k+1}_j, ε^{k+1}_{j+}. Repeat until $\varepsilon^k_i \le \varepsilon_1$ for all i = $0,\ldots,n_{k-1}$.
3. **Merge** two adjacent parts P^k_i, P^0_{i+1} into one new part P^{k+1}_j if $\varepsilon^{k+1}_j \le \varepsilon_1$. Repeat until merging is not possible.
4. **Shift** the split point shared by two adjacent parts P^k_i, P^0_{i+} to left and right while leaving the overall number of parts fixed. Keep the split that reduces the overall error, repeat until no further changes occur.

IV. RESULTS

We apply the algorithms to compute ASRs for two journeys through the environment as shown in Fig. 1. The paths of the first journey (referred to as the forward journey) are as shown in Fig. 1. The second is the return journey and the paths taken by the robot are as shown in Fig. 6 (i.e. the robot traveling in the opposite direction).

A. Results obtained for computing ASRs for paths

There are two steps in this algorithm. For the first step, we show only the results obtained for the forward journey. Figs. 7-10 show the number of surfaces selected at the four different iterations of the algorithm.

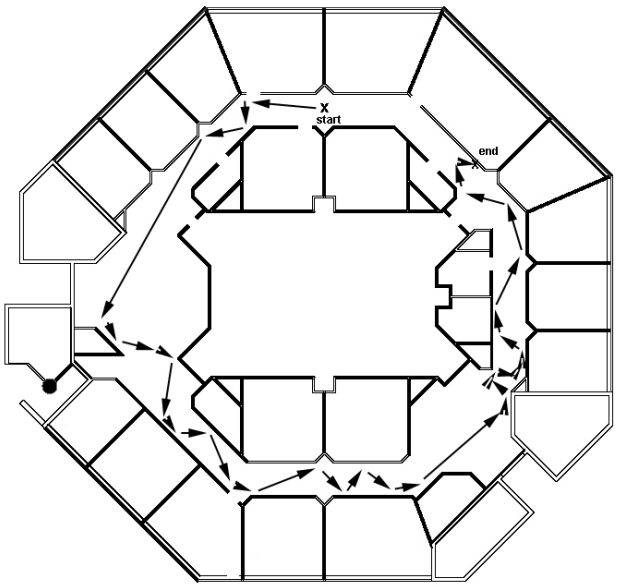

Figure 6. The paths taken by the robot in the reverse journey

Note that the boundary on the left of a path is computed independently of the boundary on the right of a path. Table 1 shows the actual number of iterations required to compute an ASR for each path taken in the forward journey. Path 1 denotes the starting path. The lesser number of iterations means that the larger surfaces perceived have provided sufficient coverage for computing the required boundary.

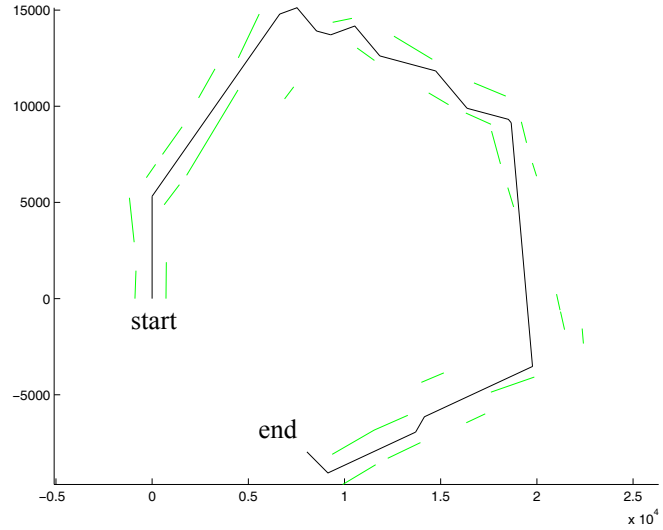

Figure 7. Surfaces (> 700mm) selected for computing ASRs for paths.

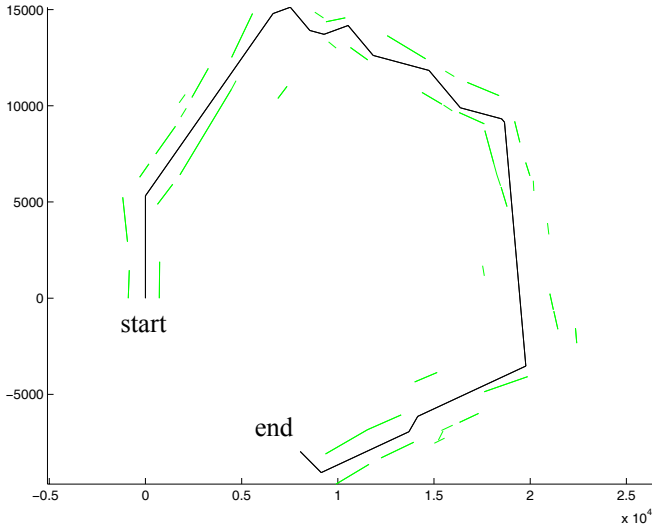

Figure 8. Surfaces (> 500mm) selected for computing ASRs for paths.

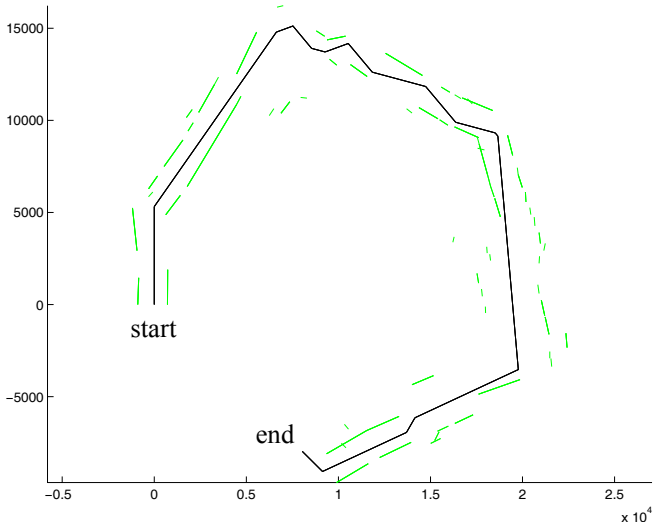

Figure 9. Surfaces (> 300mm) selected for computing ASRs for paths.

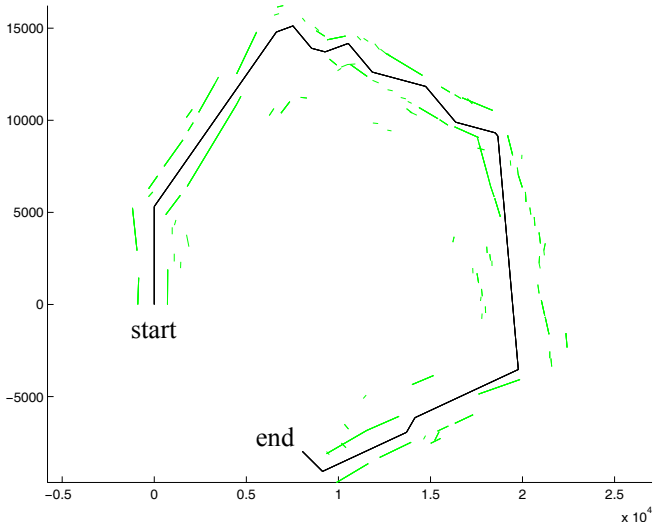

Figure 10. Surfaces (> 200mm) selected for computing ASRs for paths.

Using the surfaces obtained in the above step, Fig. 10 shows the ASRs computed for each path.

B. Results obtained for computing final ASRs

Figs. 12 and 13 show the final ASRs computed for the forward and return journeys, after applying the split and merge algorithm. Prior to split and merge, the number of ASRs created is equal to the number of paths taken to complete the journey. In the forward journey, 15 ASRs are computed. After applying the split and merge algorithm, the number of ASRs is reduced to 9 (see Fig. 12).

TABLE I. NUMBER OF ITERATIONS REQUIRED TO SELECT THE SURFACES FOR COMPUTING ASR BOUNDARY FOR EACH PATH

Paths	Left Side	Right Side
1	1	4
2	3	3
3	4	4
4	4	4
5	2	3
6	1	4
7	4	2
8	1	4
9	3	3
10	1	3
11	4	4
12	2	1
13	3	4
14	1	1
15	1	4

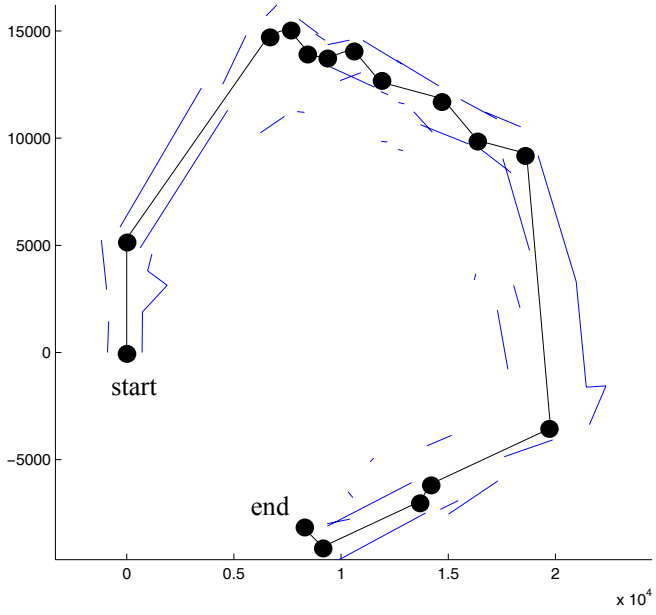

Figure 11. The ASRs computed for each path. The start and end of an ASR is indicated by a black dot. The surfaces shown denote boundaries of ASRs.

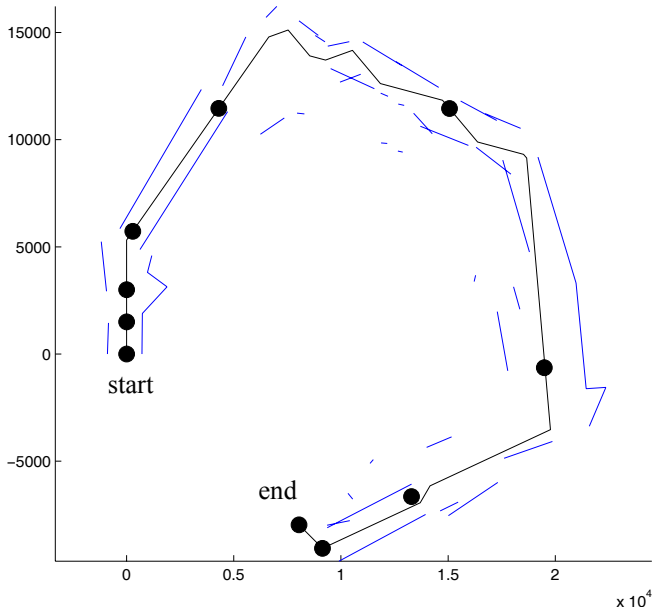

Figure 12. The final ASRs computed for the forward journey.

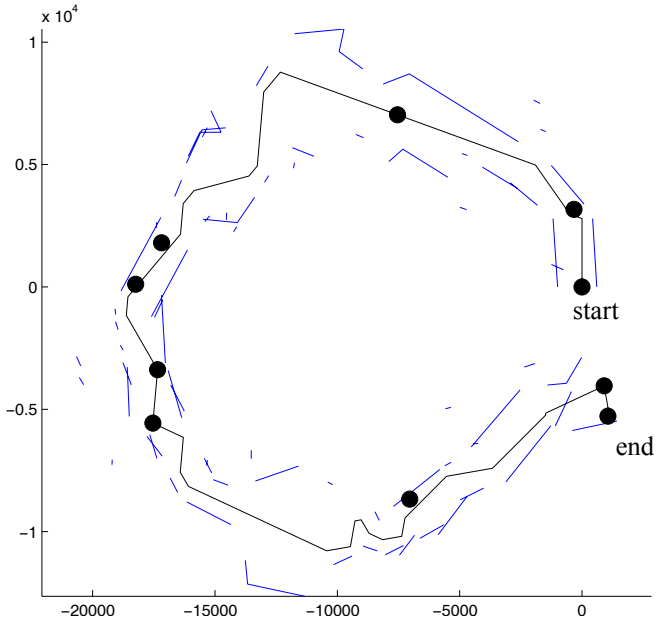

Figure 13. The final ASRs computed for the return journey.

V. Discussion

The results obtained show a network of ASRs is computed. However, it is not often the case that the split is made at a point that humans think is reasonable. However, it is interesting to observe that the same number of ASRs is produced for both the forward and return journey. This is not often the case. We have computed ASRs in the forward journey at different times and some of them showed very different ASRs are computed. This is within the expectation of the theory [1].

What do we do next? We have now successfully created a platform for testing cognitive mapping using a mobile robot with sonar sensors. Next, we want to show how such a map could be used for way-finding, a classic problem in cognitive mapping. In [9], we produced some early results for our robot to attempt finding its way home from its destination by using some commonsense reasoning strategies. We will continue to investigate how one applies commonsense reasoning to learn more about one's environment starting with a network of ASRs.

References

[1] W.K. Yeap, "Towards a computational theory of cognitive maps", *Artificial Intelligence*, vol. 34, 1988, pp. 297-360.

[2] W.K. Yeap, and M. Jefferies, "Computing a representation of the local environment". *Artificial Intelligence*, vol. 107, 1999, pp. 219-263.

[3] S. Thrun, "Learning maps for indoor mobile robot navigation", *Artificial Intelligence*, vol. 99, 1998, pp. 21-71.

[4] N. Tomatis, I. Nourbakhsh, and R. Siegwart, "Hybrid simultaneous localization and map building: A natural integration of topological and metric," *Robotics and Autonomous Systems*, vol. 44, 2003, pp. 3-14.

[5] B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli, "Local metrical and global topological maps in the hybrid spatial semantic hierarchy", in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2004, pp. 4845-4851.

[6] P. Beeson, M. MacMahon, J. Modayil, J. Provost, F. Savelli, and B. Kuipers, "Exploiting local perceptual models for topological map building", in *IJCAI-2003 Workshop on Reasoning with Uncertainty in Robotics (RUR-03),* Acapulco, Mexico, August 2003, pp, 15-22.

[7] B. Kuipers, "The spatial semantic hierarchy", *Artificial Intelligence*, vol. 119, 2000, pp. 191-233.

[8] J. Schmidt, C.K. Wong, and W.K. Yeap, "A split and merge approach to metric-topological map-building", submitted to the *18th International Conference on Pattern Recognition*, Hong Kong, August 2006.

[9] C.K. Wong, W.K. Yeap, and M. Sapiyan, "A mobile robot that maps naively but plans intelligently", in *Proc. of the Artificial Intelligence and Application Conference,* 2005, pp. 562-567.